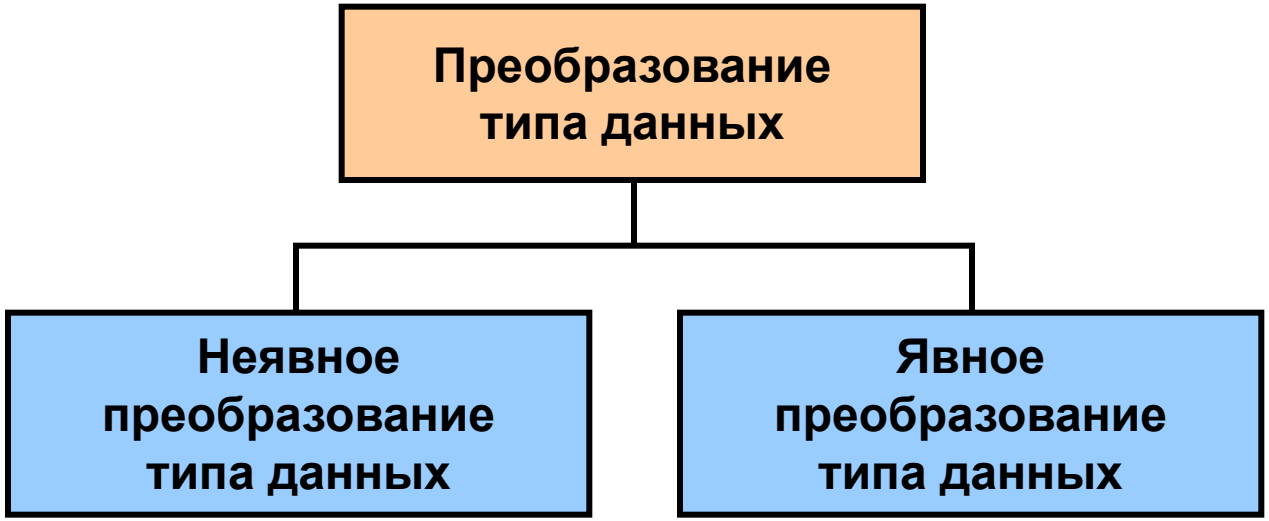
4

Использование функций преобразования и условных выражений

Рассматриваемые вопросы

- Описание различных функций преобразования, доступных в языке SQL
- Использование функций преобразования `TO_CHAR`, `TO_NUMBER`, и `TO_DATE`
- Применение условных выражений в команде `SELECT`

Функции преобразования



Неявное преобразование типов данных

Для операций присваивания Oracle может автоматически выполнять следующие преобразования:

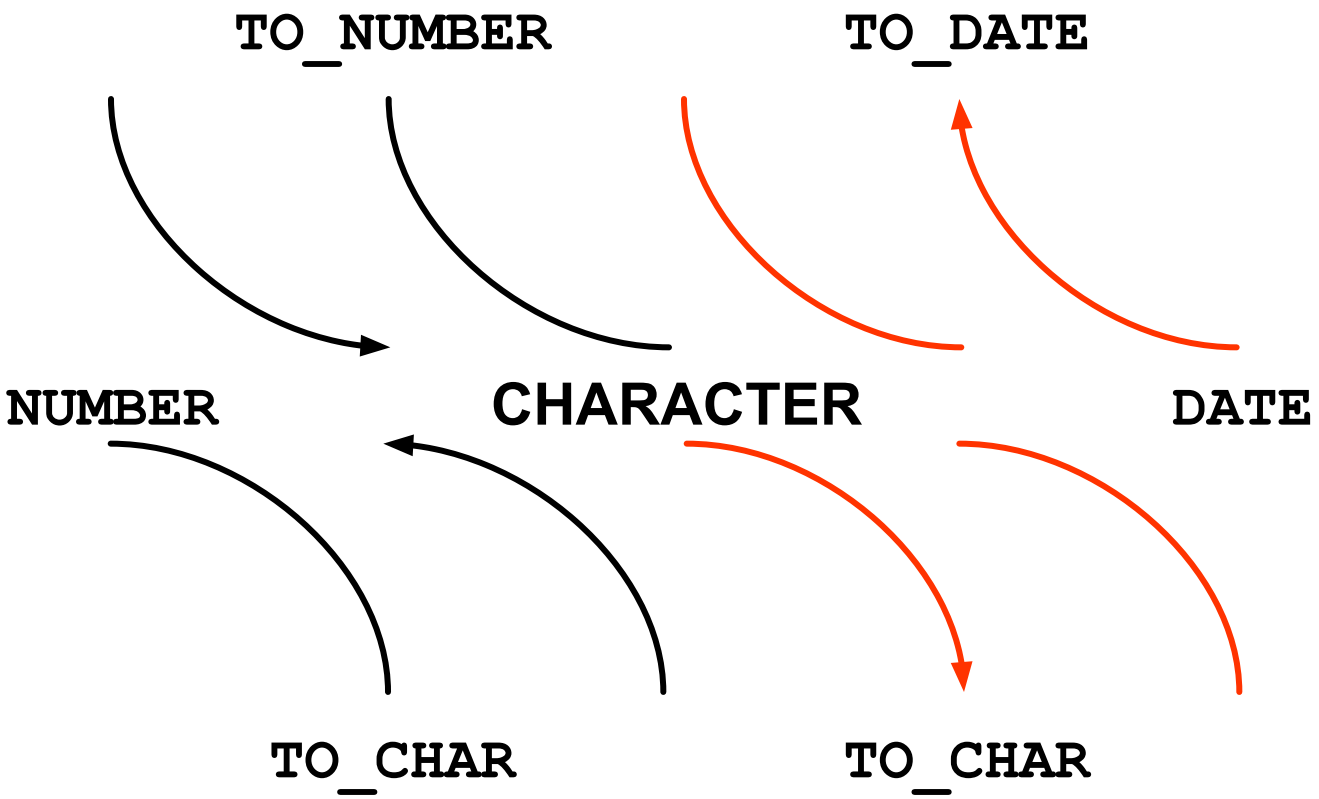
Из	В
VARCHAR2 or CHAR	NUMBER
VARCHAR2 or CHAR	DATE

Неявное преобразование типов данных

При вычислении выражений Oracle может автоматически выполнять следующие преобразования:

Из	В
NUMBER	VARCHAR2 or CHAR
DATE	VARCHAR2 or CHAR

Явное преобразование типов данных



Использование функции TO_CHAR с датами

```
TO_CHAR(date, 'модель_формата')
```

Модель формата:

- Должна быть заключена в апострофы.
- Различает символы верхнего и нижнего регистров.
- Может включать любые разрешенные элементы формата даты.
- Использует элемент *fm* для удаления конечных пробелов и ведущих нулей.
- Отделяется от значения даты запятой.

Элементы модели формата даты

Элемент	Результат
YYYY	Полный год цифрами
YEAR	Год прописью (на английском)
MM	Двузначное цифровое обозначение месяца
MONTH	Полное название месяца
MON	Трехзначное алфавитное сокращенное название месяца
DY	Трехзначное алфавитное сокращенное название дня недели
DAY	Полное название дня недели
DD	Номер дня месяца

Элементы модели формата даты

- Элементы, которые задают формат части даты, обозначающей время:
-

HH24:MI:SS AM	15:45:32 PM

- Символьные строки добавляются в кавычках:
-

DD "of" MONTH	12 of OCTOBER

- Числовые суффиксы используются для вывода числительных прописью:

ddspth	fourteenth

Использование функции TO_CHAR с датами

```
SELECT last_name,
       TO_CHAR(hire_date, 'fmDD Month YYYY')
       AS HIREDATE
FROM   employees;
```

	LAST_NAME	HIREDATE
1	Whalen	17 September 1987
2	Hartstein	17 February 1996
3	Fay	17 August 1997
4	Higgins	7 June 1994
5	Gietz	7 June 1994
6	King	17 June 1987
7	Kochhar	21 September 1989
8	De Haan	13 January 1993
9	Hunold	3 January 1990
10	Ernst	21 May 1991

...

Использование функции TO_CHAR с числами

```
TO_CHAR(число, 'модель_формата')
```

Форматы, используемые с функцией TO_CHAR для вывода числового значения в виде символьной строки:

Элемент	Результат
9	Цифра
0	Вывод нуля
$	Плавающий знак доллара
L	Плавающий символ местной валюты
.	Вывод десятичной точки
,	Вывод разделителя троек цифр

Использование функции TO_CHAR с числами

```
SELECT TO_CHAR(salary, '$99,999.00') SALARY
FROM   employees
WHERE  last_name = 'Ernst';
```

	SALARY
1	$6,000.00

Использование функций TO_NUMBER и TO_DATE

- Преобразование символьной строки в числовой формат с использованием функции TO_NUMBER:

```
TO_NUMBER(char[, 'модель_формата'])
```

- Преобразование символьной строки в формат даты с использованием функции TO_DATE:

```
TO_DATE(char[, 'модель_формата'])
```

- В этих функциях можно использовать модификатор fx. В функции TO_DATE он задает точное соответствие символьного аргумента и модели формата даты.

Пример формата даты RR

Чтобы найти сотрудников, принятых на работу до 1990 года, используйте формат RR . Выполнение команды даст одинаковый результат, независимо от того, когда выполнялась команда (сейчас или в 1999 году):

```
SELECT last_name, TO_CHAR(hire_date, 'DD-Mon-YYYY')
FROM  employees
WHERE hire_date < TO_DATE('01-Jan-90','DD-Mon-RR');
```

	LAST_NAME	TO_CHAR(HIRE_DATE,'DD-MON-YYYY')
1	Whalen	17-Sep-1987
2	King	17-Jun-1987
3	Kochhar	21-Sep-1989

Вложенные функции

- Однострочные функции могут быть вложены на любую глубину.
- Вложенные функции вычисляются от самого глубокого уровня к внешнему.

```
F3(F2(F1(столбец,арг1),арг2),
арг3)
```

Шаг 1 = Результат 1

Шаг 2 = Результат 2

Шаг 3 = Результат 3

Вложенные функции Example 1

```
SELECT last_name,
  UPPER(CONCAT(SUBSTR (LAST_NAME, 1, 8), '_US'))
FROM   employees
WHERE  department_id = 60;
```

	LAST_NAME	UPPER(CONCAT(SUBSTR(LAST_NAME,1,8),'_US'))
1	Hunold	HUNOLD_US
2	Ernst	ERNST_US
3	Lorentz	LORENTZ_US

Вложенные функции: Example 2

```
SELECT    TO_CHAR(ROUND((salary/7), 2),'99G999D99',
    'NLS_NUMERIC_CHARACTERS = '',.'' ')
    "Formatted Salary"
FROM employees;
```

	Formatted Salary
1	628,57
2	1.857,14
3	857,14
4	1.714,29
5	1.185,71
6	3.428,57

...

Общие функции

Эти функции работают с любыми типами данных и обрабатывают неопределенные значения:

- NVL (выражение1, выражение2)
- NVL2 (выражение1, выражение2, выражение3)
- NULLIF (выражение1, выражение2)
- COALESCE (выражение1, выражение2, ..., выражение*n*)

Функция NVL

Преобразует неопределенное значение в действительное:

- Используемые типы данных – DATE, символьные (CHARACTER) и числовые (NUMBER).
- Типы данных должны совпадать:
  - NVL(commission_pct,0)
  - NVL(hire_date,'01-JAN-97')
  - NVL(job_id,'No Job Yet')

Использование функции NVL

```
SELECT last_name, salary, NVL(commission_pct, 0),
    (salary*12) + (salary*12*NVL(commission_pct, 0)) AN_SAL
FROM employees;
```

1

2

	LAST_NAME	SALARY	NVL(COMMISSION_PCT,0)	AN_SAL
1	Whalen	4400	0	52800
2	Hartstein	13000	0	156000
3	Fay	6000	0	72000
4	Higgins	12000	0	144000
5	Gietz	8300	0	99600
6	King	24000	0	288000
7	Kochhar	17000	0	204000
8	De Haan	17000	0	204000
9	Hunold	9000	0	108000
10	Ernst	6000	0	72000

...

1

2

Использование функции NVL2

```
SELECT last_name,  salary, commission_pct,
       NVL2(commission_pct,
            'SAL+COMM', 'SAL') income
FROM   employees WHERE department_id IN (50, 80);
```

	LAST_NAME	SALARY	COMMISSION_PCT	INCOME
1	Mourgos	5800	(null)	SAL
2	Rajs	3500	(null)	SAL
3	Davies	3100	(null)	SAL
4	Matos	2600	(null)	SAL
5	Vargas	2500	(null)	SAL
6	Zlotkey	10500	0.2	SAL+COMM
7	Abel	11000	0.3	SAL+COMM
8	Taylor	8600	0.2	SAL+COMM

Использование функции NULLIF

```
SELECT first_name, LENGTH(first_name) "expr1",
       last_name,  LENGTH(last_name)  "expr2",
       NULLIF(LENGTH(first_name), LENGTH(last_name)) result
FROM   employees;
```

	FIRST_NAME	expr1	LAST_NAME	expr2	RESULT
1	Ellen	5	Abel	4	5
2	Curtis	6	Davies	6	(null)
3	Lex	3	De Haan	7	3
4	Bruce	5	Ernst	5	(null)
5	Pat	3	Fay	3	(null)
6	William	7	Gietz	5	7
7	Kimberely	9	Grant	5	9
8	Michael	7	Hartstein	9	7
9	Shelley	7	Higgins	7	(null)

...

Использование функции COALESCE

- Преимущество функции COALESCE по сравнению с функцией NVL состоит в том, что функция COALESCE может обрабатывать несколько альтернативных значений.
- Если первое выражение определенно, функция возвращает это выражение; в противном случает она проверяет оставшиеся выражения

Использование функции COALESCE

```
SELECT last_name, employee_id,
COALESCE(TO_CHAR(commission_pct),TO_CHAR(manager_id)
,
    'No commission and no manager')
FROM employees;
```

	LAST_NAME	EMPLOYEE_ID	COALESCE(TO_CHAR(COMMISSI...
1	Whalen	200	101
2	Hartstein	201	100
3	Fay	202	201
4	Higgins	205	101
5	Gietz	206	205
6	King	100	No commission and no manager

...

17	Zlotkey	149	.2
18	Abel	174	.3
19	Taylor	176	.2
20	Grant	178	.15

Условные выражения

- Позволяют применять логические конструкции ЕСЛИ-ТО-ИНАЧЕ (IF-THEN-ELSE) внутри команды SQL
- Два метода:
  - выражение CASE
  - функция DECODE

Выражение CASE

Помогает создавать условные запросы, которые выполняют действия логического оператора IF-THEN-ELSE:

```
CASE выражение
           WHEN сравн_выражение1 THEN возвр_выражение1
          [WHEN сравн_выражение2 THEN возвр_выражение2
           WHEN сравн_выражениеn THEN возвр_выражениеn
           ELSE else_выражение]
END
```

Использование выражения CASE

Помогает создавать условные запросы, которые выполняют действия логического оператора IF-THEN-ELSE:

```
SELECT last_name, job_id, salary,
       CASE job_id WHEN 'IT_PROG'  THEN  1.10*salary
                   WHEN 'ST_CLERK' THEN  1.15*salary
                   WHEN 'SA_REP'   THEN  1.20*salary
       ELSE      salary END     "REVISED_SALARY"
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

Функция DECODE

Помогает создавать условные запросы, которые выполняют действия логического условия CASE или оператора IF-THEN-ELSE:

```
DECODE(столбец|выражение, вариант1, результат1
                    [, вариант2, результат2,...,]
                    [, результат_по_умолчанию])
```

Использование функции DECODE

```
SELECT last_name, job_id, salary,
       DECODE(job_id, 'IT_PROG',  1.10*salary,
                      'ST_CLERK', 1.15*salary,
                      'SA_REP',   1.20*salary,
              salary)
       REVISED_SALARY
FROM   employees;
```

LAST_NAME	JOB_ID	SALARY	REVISED_SALARY
...			
Lorentz	IT_PROG	4200	4620
Mourgos	ST_MAN	5800	5800
Rajs	ST_CLERK	3500	4025
...			
Gietz	AC_ACCOUNT	8300	8300

20 rows selected.

Использование функции DECODE

Показать ставку налога на заработную плату для сотрудников 80 отдела:

```
SELECT last_name, salary,
       DECODE (TRUNC(salary/2000, 0),
                          0, 0.00,
                          1, 0.09,
                          2, 0.20,
                          3, 0.30,
                          4, 0.40,
                          5, 0.42,
                          6, 0.44,
                             0.45) TAX_RATE
FROM   employees
WHERE  department_id = 80;
```

Тест

Функция `TO_NUMBER` преобразует символьную строку или значение типа date в число, используя опциональный аргумент, определяющий модель формата.

1. True
2. False

Итоги

In this lesson, you should have learned how to:

- Изменение форматов даты для отображения
- Преобразование типов данных с помощью функций
- Использование функции `NVL`
- Использование логики `IF-THEN-ELSE` и других условных выражений в команде `SELECT`

Обзор практического занятия 4

- Создание запросов, использующий функции `TO_CHAR`, `TO_DATE` и другие
- Создание запросов, использующих условные выражения `DECODE` and `CASE`